様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２９　年　　６　月　１６　日現在

機関番号：　１２６０８
研究種目：　若手研究(B)
研究期間：　2015～2016
課題番号：　１５Ｋ２０９９５

研究課題名（和文）大規模・高精細計算を実現するGPU/CPUアプリケーション・フレームワークの開発

研究課題名（英文）Development of GPU/CPU computing framework for realizing large-scale and high-resolution simulations

研究代表者

下川辺　隆史（Shimokawabe, Takashi）

東京工業大学・学術国際情報センター・助教

研究者番号：４０６３６０４９

交付決定額（研究期間全体）：（直接経費）　　3,000,000円

研究成果の概要（和文）：格子に基づいたシミュレーションでは、広大な計算領域の場所によって求められる精度が異なる問題に有効な手法が要求されてきている。本研究では、単一のステンシル計算のソースコードをGPUとCPUで高速に実行できるフレームワーク技術、生産性の高いデータ構造、実行時パラメータを自動チューニングする機構を開発した。これを基盤とし、高精度が必要な領域を局所的に高精細にできる適合細分化格子（AMR）法をアプリケーションに簡便に適用できるAMR法フレームワークを構築した。AMR法フレームワークを圧縮性流体計算に適用し、その有用性を示した。

研究成果の概要（英文）：Recently grid-based physical simulations with multiple GPUs require effective methods to adapt grid resolution to certain sensitive regions of simulations. In this research, we have developed a framework technology that can execute the translated user code on either multiple multicore CPUs or multiple GPUs with optimization. We have also developed highly productive data structures and auto-tuning mechanism to achieve high performance on GPU. Based on these technologies, we propose a high-productivity framework for an adaptive mesh refinement (AMR) method; the AMR method is one of the effective methods on GPU to compute certain local regions that demand higher accuracy with higher resolution. The compressive fluid calculation based on the proposed AMR framework has demonstrated good results. The proposed AMR framework can contribute to hiding the complicated implementation required by the AMR method and improving the productivity of AMR applications.

研究分野：　格子法に基づいた大規模物理計算

キーワード：　高性能計算　GPU　ステンシル計算　適合細分化格子　高生産フレームワーク

１．研究開始当初の背景

研究代表者らは、様々な格子に基づいたアプリケーションを GPU で実行する研究に取り組んできた。GPU は元々は画像処理用のプロセッサであったが、ここ数年、これを汎用計算に適用する研究が盛んに行われている。GPU は従来のプロセッサである CPU と比べて計算の処理能力が非常に高く、消費電力当たりの演算性能が高いため、ペタスケールのスパコンでは GPU を大規模に搭載している。研究代表者らは、構造格子上のステンシル計算でペタスケールを達成し、GPU が極めて有用であることを示した。

GPU によるステンシル計算は高い性能が得られるものの、それにはアーキテクチャに適した高度な最適化を導入する必要がある。これを簡便に導入し生産性を高めるため、研究代表者らは、通常の C++を記述するだけで、GPU スパコンに必須な最適化をアプリケーションへ簡便に導入できる構造格子用 GPU フレームワークを提案した。

大規模 GPU 計算が可能となり、近年は広大な計算領域の場所によって求められる精度が異なる問題に有効な手法が要求されている。GPU 計算では、GPU が得意なステンシル計算を活用しながら、高精度が必要な領域を局所的に高精細にできる適合細分化格子法(AMR 法)は有効である。しかしながら、その開発は大変複雑で開発コストは高い。

２．研究の目的

研究代表者らは、アプリケーション開発者の視点で、GPU/CPU スパコン上で高精度が必要な領域をより高精細な格子で計算できる高生産フレームワークを開発する。本研究では数千台を超える GPU を搭載した (1)GPU スパコン上で局所的に高精細に計算できる適合細分化格子法(AMR 法)を確立し、局所的に 100 倍の高解像度計算を実現することを目的とする。この手法を様々なアプリケーションへ適用可能にし、様々なプロセッサで高速に実行可能にする (2) GPU/CPU 両対応した高性能・高生産 AMR フレームワークを構築する。フレームワークの開発を通して、(3) GPU および CPU で高性能な実アプリケーションを高生産に開発する技術を確立することを目指す。

３．研究の方法

本研究では、GPU/CPU スパコン上で高精度が必要な領域をより高精細な格子で計算できる高生産フレームワークを開発するために、まず (1) GPU と CPU で高性能な構造格子アプリケーションを高生産に開発するフレームワークを完成する。平行して、東京工業大学の GPU スパコン TSUBAME 上で (2)複数 GPU による適合細分化格子法(AMR 法)を開発し、局所的な領域を高精細とできる計算を実現する。そして、(2)の AMR 法に特有なデータ構造や最適化手法を取り出し、(1)の構造格子用フレームワークへ統合し、(3a) GPU/CPU 両対応した大規模計算のための AMR 法フレームワークを構築する。完成したフレームワークによって解析対象を高精細に計算する大規模アプリケーションが開発できることを検証するため、(3b) フレームワークを様々な物理問題のアプリケーションへ適用し、TSUBAME の 4,300 台の GPU や 16,000 個の CPU コアを用い大規模計算する。

４．研究成果

本研究では、まず単一のステンシル計算のソースコードを GPU と CPU で高速に実行できるフレームワーク技術を開発した。特に、生産性の高いデータ構造を導入し、実行時パラメータを自動チューニングする機構を導入することで、性能を向上させた。実アプリケーション開発に有効な、ノード間の並列化、データ入出力機構、リダクション計算を簡便に導入する基盤を構築した。次に、この GPU と CPU で高速に実行できるフレームワーク技術の中から生産性の高いデータ構造などを抽出し、GPU 上で AMR 法用のデータを効率的に柔軟に管理するデータ構造と統合した。このデータ構造を用いた様々な解像度の格子に対して、GPU 上で高速に計算する手法や袖領域のデータを交換する手法を開発した。これらを統合することで、単一 GPU 用 AMR 法フレームワークを構築した。

以下では、主な研究成果について説明する。

(1) 研究の主な成果

① 大規模 GPU/CPU 計算に向けた高生産フレームワークの概要

本研究では、初年度に、これまでの研究で開発した構造格子用 GPU フレームワークを拡張し、単一のユーザコードを GPU および CPU などの様々なアーキテクチャで高速実行する機構などを導入した高生産フレームワークを完成させた。元々のフレームワークは GPU 計算に対して高速化を行っていたが、本研究は CPU 計算において OpenMP による並列計算に対応した。また、GPU 計算では計算条件によって自動チューニングする機構を導入し、さらなる高度化に成功した。

本フレームワークは、直交格子型の解析を対象とし、各格子点上で定義される物理変数の時間変化を計算する。また、当該物理変数の時間ステップ更新は陽的であり、ステンシル計算によって行われる。実装には、ホストコードは C/C++言語、デバイスコードは CUDA を用いる。また複数 CPU および GPU 計算に対応する。プログラマは格子点上での計算についてのみ記述し、格子全体の処理はフレームワークが行う。ユーザはフレームワークを用いることで、ユーザアプリケーションにフレームワークの提供する最適化手法を簡便に適用することが可能となる。

② ステンシル計算関数の定義と実行

直交格子用 GPU/CPU フレームワークでは、ステンシル計算はフレームワークの提供する補助クラスを用い、C++ファンクタとして定義し、ステンシル計算関数と呼ぶ。拡散方程式のステンシル計算関数を示す。

```
struct Diffusion3d {
// ユーザ定義のステンシル関数（例は拡散方程式）
__host__ __device__
  float operator()(const float *f, const
  ArrayIndex &idx, float ce, float cw, float cn,
  float cs, float ct, float cb, float cc) {
    const float fn = + cc*f[idx.ix()]
  + ce*f[idx.ix(1,0,0)] + cw*f[idx.ix(-1,0,0)]
  + cn*f[idx.ix(0,1,0)] + cs*f[idx.ix(0,-1,0)]
  + ct*f[idx.ix(0,0,1)] + cb*f[idx.ix(0,0,-1)];
    return fn; // 戻り値は、ある一点の更新する値
}};
```

ステンシル計算関数は、ある格子点の隣接点を参照し、その格子点の値を更新する関数である。本フレームワークを用いることで、通常の C++を記述するだけで、同一のコードを GPU および CPU で高速に実行することに成功した。

③ 自動チューニング機構

GPU によるステンシル計算では、その性能は計算に利用するスレッド数などの実行時パラメータに大きく依存する。本フレームワークでは、CUDA のブロック内の x と y 方向のスレッド数と z 方向にマーチングする格子数をチューニングするパラメータとする自動チューニング機構を新たに導入した。自動チューニングは、プログラムの実行中に行われる。ステンシル計算関数の実行毎に新しい自動チューニング・パラメータが指定され、その実行時間が計測される。全てのパラメータで計測を完了すると、それ以降のステンシル関数の実行では、実行時間を最短とする最適なパラメータを用いる。チューニング中は性能低下が見られるが、一般的に格子を用いた計算では、ステンシル関数は数万回を超え実行されるため、チューニング・パラメータ決定のための性能低下はアプリケーション全体の実行時間と比較すると無視することができる。

図 1 は単一 GPU で自動チューニング機能を用いた場合と代表的なスレッド数、分割数に固定した場合の拡散計算の実行性能を示す。図に示すように、自動チューニングを取り入れた計算は、全ての格子サイズの領域に対して高速な実行速度を示し、導入した自動チューニング機構が有効であることを示した。

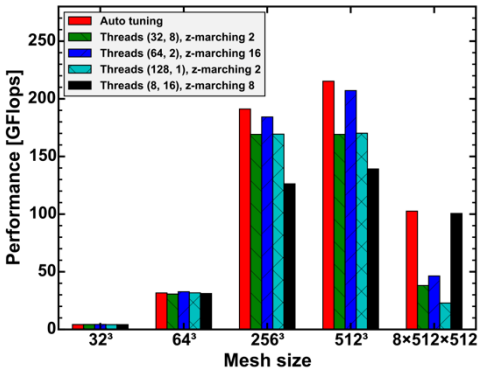

図 1 単一 GPU における自動チューニング機能を用いた拡散計算の実行性能

④ AMR 法フレームワークの概要

開発した大規模 GPU/CPU 計算に向けた高生産フレームワークを基盤とすることで、局所的に高精細に計算できる適合細分化格子法(AMR 法)を適用できる AMR 法フレームワークを構築した。本フレームワークは、直交格子型の解析を対象とし、各格子点上で定義される物理変数の時間変化を計算する。当該物理変数の時間ステップ更新は陽的であり、ステンシル計算によって行われる。各時間ステップにおける格子の解像度は局所的に変化する。本フレームワークは直交格子上にブロック領域を定義し、その領域内を再帰的に細分化するブロック AMR 法として実装する。ブロック AMR 法では、計算領域内に様々な解像度のブロックが存在するが、ユーザは単一解像度の格子点上での計算についてのみ記述し、格子全体の処理、解像度の変更、解像度の異なるブロック間での袖領域のデータ交換などはフレームワークが行う。木構造を意識しないプログラミングモデルの構築に成功した。

⑤ AMR 法フレームワークのデータ構造

本フレームワークの対象とする AMR 法では構造格子を再帰的に細分化し、その空間的配置を木構造で表す。木構造の各リーフノードには、一つの格子ブロックを割り当てる。格子ブロックは典型的には 2 次元で 16×16 格子程度である。汎用的に用いることができるフレームワークとするため木構造中には格子ブロックの実データを保持しない。メモリ空間で多数の格子ブロックをフラットに配置し、リーフノードとメモリ上の格子ブロックの間は整数値で対応づける。図 2 に複数の格子ブロックの物理空間配置とメモリ空間の配置の模式図を示す。3 次元計算では八分木、2 次元空間では四分木となる。

木構造は空間的に点対称な広がりを持つため、単一の木構造で計算対象とする物理空間を充填させると、計算対象の形状によっては無駄な領域が発生する。そこで、図 3 のように複数の木構造を物理空間に配置する。これによって任意の領域形状を柔軟に表現することを可能とした。

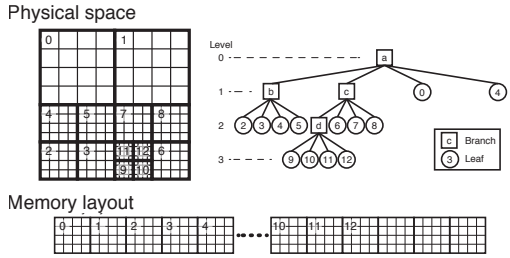

図 2　格子ブロックの木構造による物理空間配置とメモリ空間配置

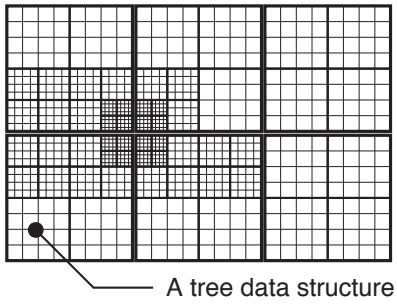

図 3　複数の木構造によるデータ構造

⑥ AMR 法におけるステンシル計算関数の定義と実行

各リーフノードに関連づけられた格子ブロックは隣接格子のデータを保持する袖領域を持つこととすることで、ステンシル計算中に異なる解像度の格子点を参照しない方法を採用した。この方法は多少の性能低下が予想されるものの、構造格子用フレームワークと同一のステンシル計算関数を用いることができるため、構造格子用コードを簡単に AMR 法用コードへ移植可能となり、生産性が高い。

ステンシル計算の実行は、図 2 に示した多数の配列を保持できるデータ構造に対して実行する。格子ブロックをメモリ上でフラットに配置することで、GPU は効率的に計算することが可能である。これによって高い性能を達成した。

⑦ AMR 法における 2 格子ブロック間の袖領域データ交換

AMR 法フレームワークでは、リーフノード上の格子ブロックは、ステンシル計算のために袖領域を持つ。時間ステップを進めるためには、異なる解像度間で袖領域のデータ交換が必要である。低解像度の格子データから高解像度の格子データを生成する場合、新たに挿入される格子点には補間された値を代入する。この値は 1 次関数により補間される。

袖領域のデータ交換は次の順に行う。まず、(1) 値の補間が必要ない同一解像度の格子ブロック間で袖領域のデータ交換を行う。次に (2) 値の補間が必要ない高解像度格子ブロックから低解像度格子ブロックへ袖領域のデータを転送する。格子点上に値が定義されている場合は、高解像度格子の値を間引くことで低解像度格子の値を決定する。最後に (3) 値の補間を必要とする低解像度格子ブロックから高解像度格子ブロックへ袖領域のデータを転送する。(3) では、補間のために隣接格子点を参照するため、先に (1)、(2) を実行する。袖領域の交換を GPU 上で効率的に行うため、(1), (2), (3) の処理をそれぞれ一つの CUDA カーネルとして実装する。リーフノード間の隣接関係をテーブルとして保持することで高速化に成功した。

⑧ 格子ブロックの解像度の変更

リーフノードの持つ格子ブロックの解像度の変更は、物理量の変化等に着目する必要があるため、フレームワーク側では自動的には行わず、フレームワークは変更を指示する関数を提供する。

ユーザがある格子ブロックをより高い解像度にする、すなわちレベルを上げるよう指示すると、そのブロックは強制的に一段レベルが上がる。隣接するブロックと 2 段以上のレベル差ができた場合は、隣接するブロックのレベルを 1 段上げる。これを繰り返す。一方、ユーザがある格子ブロックをより低い解像度にするよう指示すると、隣接する格子が同じレベルか 1 段下のレベルの時のみ、1 段下げる。隣接ブロックに高い解像度の格子がある場合、物理的にその近傍で高い解像度が必要なことが示唆されているため、解像度は下げない。解像度の変更においても GPU で効率的に実行するため、解像度の変更前後のリーフノードの関係をテーブルとして保持する。全ての格子ブロックはプールで管理される。解像度変更時に新しい格子ブロックが必要な際には、そのプールから取得され、解像度の変更後に不要となった格子ブロックはそのプールに返却される。これによってメモリを効率的に使用することを実現した。

⑨ AMR 法フレームワークの圧縮性流体計算への適用例

本研究では、移流計算や圧縮性流体に本フレームワークを適用した。ここでは、実装が複雑な問題における本フレームワークの有効性を評価するため、3 次元オイラー方程式による圧縮性流体計算への本フレームワークの適用例を示す。この計算では密度、速度、エネルギーに関する 5 つの変数を扱う。移流項は、2 次元 3 次精度風上手法 に 3 次 TVD ルンゲ・クッタ法を用いる。このように解像度の変更を複数の変数に対して同時に行う必要があり、管理が複雑である。

図 4 に、この計算コードで計算したレイリーテイラー不安定性の計算結果を示す。図の格子状の水色線は各リーフノードが持つ格子ブロックを表す。一つの格子ブロックは 20×20 格子で，5 レベルの AMR を用いる。流体の色は二つの異なる密度を表している。高解像

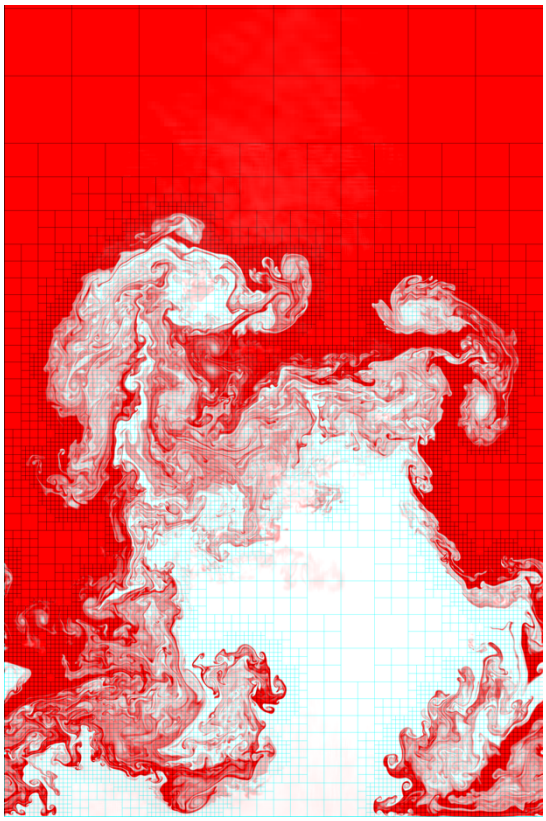

図 4 AMR 法を用いたレイリーテイラー不安定性の計算結果

度が必要となる界面を含む領域を AMR 法で高解像度にしている。非物理的な振動を発生せずに計算することに成功した。

(2) 得られた成果の国内外の位置づけとインパクト

本研究では、大規模 GPU/CPU 計算に向けた高生産フレームワークを完成させ、それを発展させた局所的に高精細に計算できる適合細分化格子法(AMR 法)を導入した単一 GPU 用のフレームワークを構築した。

ペタスケールのスパコンでは、低消費電力かつ高性能を達成するため数千台を超える GPU が搭載され、日本、米国、中国などで稼働している。格子計算はスパコンを利用する代表的なアプリケーションで、局所的に高精細とした計算を GPU で実現できた意義は大きい。

GPU アプリケーションは、開発コストが高いだけでなく、CPU では動作せず、生産性や保守性が低い。さらに実装が複雑となる AMR 法を導入した GPU アプリケーションの開発コストは極めて高い。本研究で開発した大規模 GPU/CPU 計算に向けた高生産フレームワークは、機種固有のコードの差異を吸収し、単一の計算コードから GPU または CPU で実行できるコードを生成することが可能である。AMR 法フレームワークでは、局所的に高精細にできる GPU アプリケーションを高い生産性で開発することが可能であることを示した。大規模 GPU/CPU 計算に向けた直交格子用フレームワークおよび AMR 法フレームワークの開発を通して、様々なアーキテクチャで動作する高性能アプリケーションを高生産に開発する方法について有益な知見が得られた。

５．主な発表論文等

（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 1 件）

① Takashi Shimokawabe, Takayuki Aoki, and Naoyuki Onodera, ″High-productivity Framework for Large-scale GPU/CPU Stencil Applications,″ IHPCES/ICCS 2016, San Diego, USA, June 2016.査読有

〔学会発表〕（計 9 件）

① Takashi Shimokawabe, ″Large-scale GPU Applications Based on a High-productivity Stencil Framework,″ Computational Sciences Workshop 2017, 湘南国際村センター（神奈川県・葉山）, March 6, 2017. (招待講演)
② Takashi Shimokawabe, Toshio Endo, Naoyuki Onodera, Takayuki Aoki, ″Performance Evaluation of Wind Simulation Based on a GPU-computing Framework to Realize Large-scale Stencil Computations Beyond Device Memory Capacity,″ The 7th AICS International Symposium, 神戸大学（兵庫県・神戸）, February 23, 2017.
③ 下川辺隆史, 遠藤敏夫, 青木尊之, ″GPU デバイスメモリを超える計算を可能とするためのステンシル計算フレームワークの拡張とその性能評価″, 日本計算工学会 第 21 回計算工学講演会, 朱鷺メッセ：新潟コンベンションセンター（新潟県・新潟）, 2016 年 6 月 1 日.
④ Takashi Shimokawabe, ″Large-scale GPU-based CFD Applications based on a High-productivity Stencil Framework,″ Parallel CFD 2016, 神戸国際会議場（兵庫県・神戸）, May 10, 2016. (招待講演)
⑤ Takashi Shimokawabe, ″Advanced High-Productivity Framework for Large-Scale GPU/CPU Stencil Computations,″ GTC 2016, San Jose, CA, USA, April 4, 2016. (GTC Poster Award finalist)
⑥ Takashi Shimokawabe, ″High-resolution Weather Prediction Code based on High-productivity Framework for Multi-GPU computation,″ 2nd Annual Meeting on Advanced Computing System and Infrastructure (ACSI2016), 九州大学（福岡県・福岡）, 2016 年 1 月 19 日.
⑦ Takashi Shimokawabe, Takayuki Aoki, and Naoyuki Onodera, ″Advanced High-productivity Framework for Stencil Applications on GPU Supercomputers,″ 3rd International Workshops on Advances in

Computational Mechanics, KFC Hall & Rooms(東京都・両国), October 13, 2015.（招待講演）
⑧ 下川辺隆史, 青木尊之, 小野寺直幸, ″自動チューニング機構の導入によるステンシル計算のための GPU コンピューティング・フレームワークの高度化″,日本応用数理学会 2015 年 年会, 金沢大学(石川県・金沢), 2015 年 9 月 11 日.
⑨ 下川辺隆史, 青木尊之, 小野寺直幸, ″ステンシル計算のための高生産 GPU コンピューティング・フレームワークの高度化″, 日本計算工学会 第 20 回計算工学講演会, つくば国際会議場（茨城県・筑波）, 2015 年 6 月 8 日.

〔その他〕
ホームページ等
http://www.sim.gsic.titech.ac.jp/Japanese/Member/shimokawabe/index-ja.html

６．研究組織
(1)研究代表者
下川辺　隆史（SHIMOKAWABE, Takashi）
東京工業大学・学術国際情報センター・助教
研究者番号：40636049